Jurnal
Teknologi

Full Paper

FUSION SPARSE AND SHAPING REWARD FUNCTION IN SOFT ACTOR-CRITIC DEEP REINFORCEMENT LEARNING FOR MOBILE ROBOT NAVIGATION

Mohamad Hafiz Abu Bakar[a], Abu Ubaidah Shamsudin[a]*, Zubair Adil Soomro[a], Satoshi Tadokoro[b], C. J Salaan[c]

[a]Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia
[b]Tohoku University, 2 Chome-1-1 Katahira, Aoba Ward, Sendai, Miyagi 980-8577, Japan
[c]Department of Electrical Engineering and Technology, MSU-Iligan Institute of Technology, Andres Bonifacio Ave, Iligan City, 9200 Lanao del Norte, Philippines

Article history
Received
4 April 2023
Received in revised form
24 August 2023
Accepted
2 October 2023
Published Online
18 February 2024

*Corresponding author
ubaidah@uthm.edu.my

Abstract

Nowadays, the advancement in autonomous robots is the latest influenced by the development of a world surrounded by new technologies. Deep Reinforcement Learning (DRL) allows systems to operate automatically, so the robot will learn the next movement based on the interaction with the environment. Moreover, since robots require continuous action, Soft Actor Critic Deep Reinforcement Learning (SAC DRL) is considered the latest DRL approach solution. SAC is used because its ability to control continuous action to produce more accurate movements. SAC fundamental is robust against unpredictability, but some weaknesses have been identified, particularly in the exploration process for accuracy learning with faster maturity. To address this issue, the study identified a solution using a reward function appropriate for the system to guide in the learning process. This research proposes several types of reward functions based on sparse and shaping reward in SAC method to investigate the effectiveness of mobile robot learning. Finally, the experiment shows that using fusion sparse and shaping rewards in the SAC DRL successfully navigates to the target position and can also increase accuracy based on the average error result of 4.99%.

Keywords: Soft Actor Critic Deep Reinforcement Learning (SAC DRL), Deep Reinforcement Learning, Mobile robot navigation, Reward function, Sparse reward, Shaping reward

Abstrak

Pada masa kini, kemajuan dalam robot autonomi adalah dipengaruhi oleh perkembangan dunia terkini yang dikelilingi dengan teknologi baharu. “Deep Reinforcement Learning” (DRL) membolehkan sistem beroperasi secara automatik, justeru robot akan mempelajari pergerakan seterusnya berdasarkan interaksi dengan persekitaran. Selain itu, disebabkan robot memerlukan tindakan berterusan, “Soft Actor Critic Deep Reinforcement Learning” (SAC DRL) dianggap sebagai penyelesaian pendekatan DRL yang terkini. SAC digunakan kerana ia boleh mengawal tindakan berterusan untuk menghasilkan pergerakan yang lebih tepat. Tambahan pula, asas SAC adalah teguh terhadap ketidakpastian, tetapi terdapat beberapa kelemahan telah dikenal pasti, terutamanya dalam proses penerokaan untuk pembelajaran ketepatan dengan kematangan yang cepat. Bagi menangani isu ini, kajian mengenal pasti penyelesaian terkini menggunakan fungsi ganjaran yang sesuai untuk sistem sebagai panduan di dalam proses pembelajaran. Penyelidikan ini mencadangkan beberapa jenis fungsi ganjaran berdasarkan ganjaran jarang dan membentuk di dalam kaedah SAC untuk menyiasat keberkesanan pembelajaran robot mudah alih. Akhir sekali, eksperimen menunjukkan bahawa penggunaan gabungan ganjaran jarang dan membentuk di dalam SAC DRL berjaya menavigasi ke kedudukan sasaran dan juga boleh meningkatkan ketepatan berdasarkan hasil purata ralat sebanyak 4.99%.

Kata kunci: *Soft Actor Critic Deep Reinforcement Learning* (SAC DRL), *Deep Reinforcement Learning*, Robot mudah alih navigasi, Fungsi ganjaran, Ganjaran jarang, Ganjaran membentuk

1.0 INTRODUCTION

With the evolution of the modern era nowadays, the use of artificial intelligence-based technology has emerged as a technical breakthrough that can improve the conventional method [1, 2], which is a trend that is widely used in robot control. There are various studies carried out by previous researchers involving artificial intelligence especially robot control by using Reinforcement Learning [3, 4] and there are different technologies that have been introduced to mobile robot applications for the navigation process. Most of the studies conducted have indicated the trends of some researchers in new technologies that have been implemented to improve the mobile robot abilities using machine learning [5-7].

Machine Learning is one of the effective branches of AI (Artificial intelligence), which is practically used for training the robot without complete human supervision [8]. Reinforcement Learning enables the agent (robot) to learn the next action based on the interaction between the agent and the environment, allowing the system to run and control autonomously. Recently, there are many algorithms on Reinforcement learning among which are Q-Learning [9, 10], SARSA [11], DQN [12], DDPG [13], and SAC [14]. There is no doubt that these will continue to make headlines in the coming years.

In RL, there are two methods: on-policy and off-policy [15], with the off-policy becoming the preferred method because it can save the learning process in the replay memory [12]. Furthermore, replay memory can improve the learning process effectively based on learning factors based on experience. At the same time, the off-policy is divided into two states consisting of deterministic and stochastic policies. Deterministic is a method based on no probability of action selection. In contrast, Stochastic policy is based on randomness probability action, which allows the agent to explore further and improve its performance.

Recently, SAC is state of the art in RL which is based on a stochastic policy that applies maximum expected reward and entropy in the exploration process. According to this article [14, 16], SAC still has a lot of room for improvement, particularly in learning accuracy and maturity, which can be attributed to the correct and accurate use of the reward function. Otherwise, using the inappropriate reward function can also make the system development process more complex and extensive due to the trial and error process in determining the agent's ability to learn effectively and quickly. Our motivation in conducting this research can improve the SAC development process, particularly the structure of the reward function and can speed up the robot's learning process in terms of maturity for a learning curve.

To overcome this problem, we suggest producing a reward function based on fusion sparse and shaping reward in adapting the robot to learn better and mature quickly. The main reason for using sparse and shaping is to maximize the benefits of the different reward type functions. Furthermore, the SAC method's robot learning process can be improved by introducing fusion sparse and shaping reward functions.

2.0 RELATED WORK

In practice, DRL is a system that is one of the autonomous system concept's characteristics. Numerous autonomous applications in this sector implemented DRL with varied uses [16-19], especially the autonomous mobile robot in this study that is based on navigation task [6, 7, 20]. There are two types of policies commonly used and practiced in DRL [15, 21], generally on-policy and off-policy. Both have different policies and practices to how the system is used. The concept of on-policy refers to a method of behaviour policy that seeks to evaluate or improve policies used to make decisions [15]. For off-policy methods, the policy is to improve policies that are not used to generate data and allow the agent to explore the environment without adhering strictly to policy [15]. Off-policy users are generally also due to its main feature, which practices using replay memory, which is very useful in the learning process and assisting in performance [12]. Off-policy usually includes two methods consisting deterministic and stochastic especially using for continuous action. This paper [14] demonstrates the effectiveness of a stochastic policy that significantly helps the system predict the following action with more significant variance and is effective in exploration. Furthermore, using a stochastic policy can reduce the repeated selection of actions leading to overestimation in the system network in DRL algorithms [22].

The type of control action is also an important factor in determining the appropriate algorithm user in the robot system. There are two types of control actions for this [15] consisting of the continuous action [23] and discrete action [12]. The use of continuous action is main as an important indicator in the solution's selection since the accuracy of the navigation process is required in determining the performance of this system. Based on this paper [6], a comparative experiment in mobile robot navigation uses two types of methods from the same action category, DPPG, and SAC. Based on the results, SAC has produced better performance on the simulation results. The concept of SAC is explained in this work [14], where stochastic policy is used as an actor policy for the agent to learn. SAC maximises the use of entropy when making decisions. This paper also demonstrates the state-of-the-art performance obtained by implementing the SAC in robot applications based on continuous action hence making it is an excellent option for the architecture algorithm in this study. SAC also has a weakness that can complicate the exploration process, particularly the learning policy without pre-learning, that makes learning challenging and takes a long time to the maturity learning [16]. The solution for this issue is to implement an appropriate reward function structure that allows the system to learn more easily and quickly, increasing the overall system's efficiency [24].

Previous research has identified various types of reward functions, which are classified into two reward categories: sparse and non-sparse (shaping reward). The Sparse reward is a binary reward after entire episodes rather than continuous feedback after each action. Some processes, that are discussed in the articles [25, 26] require sparse reward in determining the exploration of the environment and guiding the robot in system control. Sparse rewards will also simplify and reduce the complexity of the robot's exploration process in the environment [26]. Article in [27] proposed Scheduled Auxiliary Control (SACX) to simulate robot manipulation for the exploration process in the environment by modifying sparse rewards, but it has not yet been applied in the real world. In fact, [26] demonstrates the benefits of sparse reward to the UAV navigation system by using DRL with nonexpert helpers in a large-scale environment. This paper [25] investigates an alternative by employing sparse reward RL in video game environments and it has successfully overcome the sparse reward system's weakness through a combination of curiosity-driven exploration and unsupervised auxiliary tasks.

In some previous research papers on non-sparse(shaping) rewards from article [16], the SAC algorithm is used to control an autonomous underwater vehicle (AUV) and has introduced a comprehensive external reward function to overcome the weakness of sparse reward. In [28], which addresses a limitation of the Partially Observable Markov Decision Process (POMDP) for UAV navigation and it is solved based on the non-sparse reward function and introduces the specific reward function based on domain knowledge. Yiqiu Hu [29] released a paper in 2021 that used federated reinforcement learning by implementing reward shaping by experimenting on GridWorld and successfully demonstrated a system that can improve the efficiency and quality of training. In the paper [30], the reward horizon (optimal bound for reward function) is used in the MDP RL system to conduct experiments on the shaping reward to increase the speed of the training process. Articles [24, 26] have conducted experiments by increasing the speed of the training process through the shaping rewards into the RL algorithm implementation. [24] presented an additional subtask reward function in the DRL policy to control the manipulator for the purpose of controlling the trajectory movement. These rewards can help the agents to learn faster and to reduce mistakes in exploiting the environment. In [31], an experiment was conducted to solve the sparse reward problem by developing a reward function based on the knowledge domain (shaping reward) for kinematic mobile robots using improved DQN.

In summary, the use of an efficient reward function should be highlighted throughout system development, particularly in the RL algorithm. According to the previous papers [31, 32] combining both types of reward functions can also assist the system in learning tasks easily and effectively. Several previous papers [28, 30, 33] have successfully demonstrated that the reward function based on shaping can accelerate the learning process. Therefore, the purpose of conducting this experiment is to apply the concept of an effective reward function based on a combination of sparse and non-sparse (shaping reward) to overcome the weaknesses that exist in the reward function of the SAC algorithm [16], that can help agents learn correctly while also accelerating the learning process in terms of maturity to adapt the environment.

3.0 METHODOLOGY

The study's main objective is to develop a SAC control system that will control the robot without specific instructions based on a suitable reward function that can assist learning and improve navigation accuracy. MATLAB and Simulink are used extensively in this system's development training and simulation phases. Several types of reward functions derived from a combination of sparse and shaping reward were designed to evaluate the significance and effectiveness of this study.

3.1 SAC Agent Development

Based on Figure 1 shows the block diagram developed in the simulation. The navigation input is based on the location's initial and target values. Then, the system will operate and be controlled by the SAC agent based on the trial-and-error concept. Through the process of control using SAC, the system will learn randomly and generate action for the next movement as an output of the system. Lastly, the system will produce observations and rewards to be used as the next input to the controller system for learning and exploring in the environment.

Divided into four categories: SAC controller system, mobile robot, environment, and mobile robot interaction with the environment. Firstly, the SAC controller system has input based on observation (current position robot, Lidar ray, target position, difference distance, and total difference), and the output is action (linear and angular velocity). Next, the SAC system provides linear and angular velocity input to the mobile robot, which produces an output that includes the robot's current position and Lidar ray reading while the environment only outputs the target position (x, y, theta). Finally, mobile robot interaction with the environment generates response output based on the initial and target distances (x, y, theta) and the total difference.

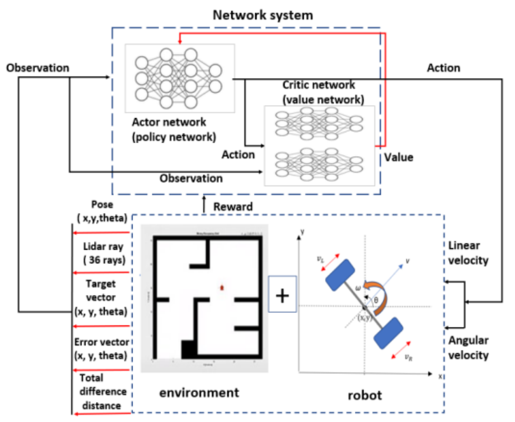

Figure 1 Block diagram of the system development

A. SAC Agent

SAC is a model-free algorithm with applies the off-policy method [14] in the learning process. In addition, SAC agents use actor and critic as function approximators to select the next action. In order to develop the agent and train the system to perform actions based on the critic and actor functions, the rlSACAgent (agent function) [34] is utilized with the specified actor and critic, as depicted in Equation (1).

$$agent \;=\; rlSACAgent\;(actor, critic) \qquad (1)$$

In this research, the implementation includes the utilization of two critics to enhance the stability of the optimization network [14]. Consequently, the agent is created within a simulation, as represented by Equation (2).

$$\text{agent} \;=\; \text{rlSACAgent (actor, [critic1 critic 2])} \qquad (2)$$

To implement the function of the reinforcement learning concept using Simulink as in Figure 2, several features such as observation, action, reward, and isDone condition must be developed in this work.

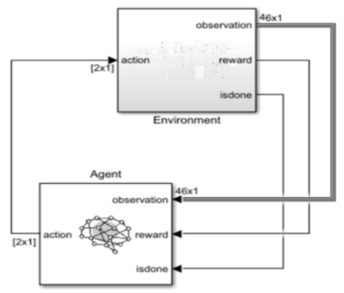

Figure 2 Block of Simulink used in Reinforcement Learning

B. SAC Neural Network Structure

The neural network in the SAC is designed to increase the system's intelligence and robustness. In SAC, there are two parts of network, first is for the actor network and the second part is the critic network.

According to Figure 3, the network's output follows a Gaussian distribution, indicating that the action is generated by probability distribution. The Gaussian distribution layer is constructed by combining the standard deviation and mean to determine the action value with high precision.

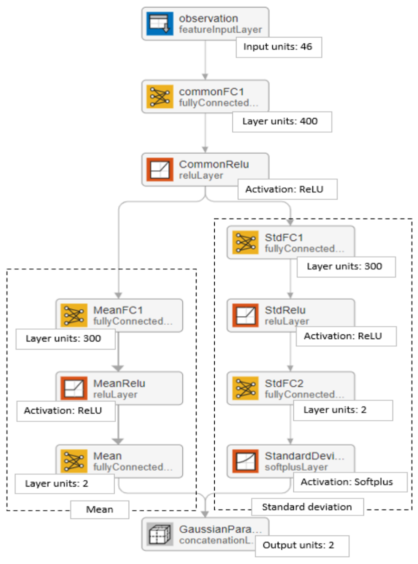

Figure 3 Flow chart of actor network system by using the Deep network designer application

In Figure 3, the actor network has a single input that utilizes the features input layer, which contains numeric scalar data. The network layer connector is implemented using the full connector layer, and the activation layer for the hidden layer is set as the ReLU layer. At the end of the standard deviation section, the softplus layer is used exclusively as an activation layer.

In this research, the full connector layer is preferred over the convolution layer because it can leverage all the features from the previous layer's combination [35]. The ReLU activation function layer is extensively used throughout the network to prevent simultaneous activation of all neurons and reduce training time [36]. As shown in Figure 3, the softplus activation function layer is employed in the standard deviation section to ensure positive output values. Moreover, the softplus layer, being a smoother continuous version of the ReLU layer, also generates a Gaussian policy in the actor network.

In the meantime, the critic network is designed to take input from observations and actions. To improve decision-making stability and efficiency, the critic network is updated using two networks. It's worth noting that the structure of the critic network remains the same for both critic 1 and critic 2 networks in this research. Both network structures used similar designs are illustrated in Figure 4. Specifically, the critic network utilizes the feature input layer (input layer), the linear function layer using the fully connected layer as the connector for all and the ReLU activation function layer as the hidden layer. The output of this network is then used as input for the actor-network. The critic network plays a crucial role by providing criticisms or comparisons, and its values are continuously updated to improve the decision-making process in subsequent steps.

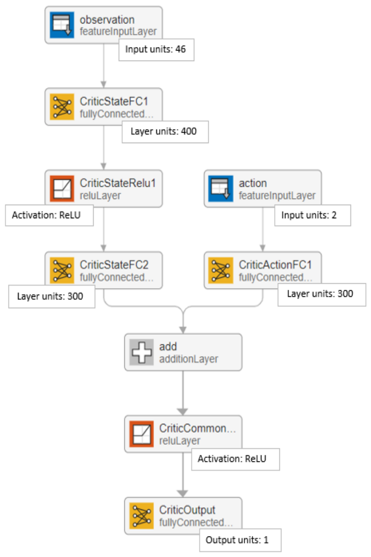
observation featureInputLayer
Input units: 46
CriticStateFC1 fullyConnected...
Layer units: 400
CriticStateRelu1 reluLayer
Activation: ReLU
action featureInputLayer
Input units: 2
CriticStateFC2 fullyConnected...
Layer units: 300
CriticActionFC1 fullyConnected...
Layer units: 300
add additionLayer
CriticCommon... reluLayer
Activation: ReLU
CriticOutput fullyConnected...
Output units: 1

Figure 4 Flowchart of first or second critic network system by using the Deep network designer application

C. Observation Space

In this topic, observation space (observation signal) is an input to the network system that the agent uses to learn how to improve the system to increase the agent's efficiency in controlling navigation instructions to the robot. This robot uses 46 observation signal inputs in total, including position (x, y, theta), Lidar ray (36 rays), target position (x, y, theta), error vector (x, y, theta), and total difference. As illustrated in Figure 5, these signals were used for the observation feature.

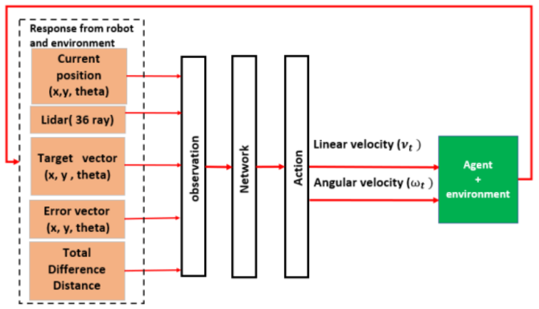
Response from robot and environment
Current position (x,y, theta)
Lidar(36 ray)
Target vector (x, y , theta)
Error vector (x, y, theta)
Total Difference Distance
observation
Network
Action
Linear velocity (v_t)
Angular velocity (ω_t)
Agent + environment

Figure 5 Block diagram of the system development

For the observation space part, the agent utilised the robot's current position and target position to determine movement and distance differences for the navigation. The total distance difference is calculated as follows in Equations (3)-(7).

Equation (3) and Equation (4) describes an error for each of the two axes, x and y. The value will be updated as an observation parameter in the differential between the actual and the target position.

$$E_x = (c_x - t_x) \tag{3}$$

and,

$$E_y = (c_y - t_y) \tag{4}$$

where E_x is the error vector axis -x, E_y is the error vector axis -y, c_x is the current axis-x, c_y is the current axis-y, t_x is the target axis-x and t_y is the target axis-y.

The error vector theta(θ) is differently calculated compared with vector-x, y because it is a rotation of axes depending on the simulation platform. Table 1 describes the error θ value based on their condition criteria to estimate the value of positioning error for the body robot.

Table 1 Error theta (θ) value based on their condition criteria

Case	Angle error condition	Error value
1	$\theta_c > 0$ && $\theta_t > 0$	$\theta_t - \theta_c$
2	$\theta_c > 0$ && $\theta_t < 0$	$-\theta_t + (\pi - \theta_c)$
3	$\theta_c < 0$ && $\theta_t > 0$	$\theta_t + (\pi - \theta_c)$
4	$\theta_c < 0$ && $\theta_t < 0$	$\theta_t - \theta_c$
5	$\theta_c = = 0$	θ_t

where θ_c is the current theta(θ) and θ_t is the target theta(θ).

$$D_{ct} = \sqrt{((c_x - t_x)^2 + (c_y - t_y)^2)} \tag{5}$$

and,

$$D_{ti} = \sqrt{((t_x - i_x)^2 + (t_y - i_y)^2)} \tag{6}$$

where i_x is initial axis-x and i_y is initial axis-y. D_{ct} is the value for the distance between current to target position and D_{ti} is the value for the distance from the initial target to the target position.

Both of value used to determine total difference distance (D_T) can be written as follow Equation (7).

$$D_T = D_{ti} - D_{ct} \tag{7}$$

During the navigation phase, three-movement parameters were used: axis-x, axis-y, and theta (orientation angle of body robot). Figure 6 demonstrates two different situations for the robot's orientation angle. The structure of the angle rotation ratio is important for assisting in the development of the system more effectively, and that concept was also used in designing the angle correction reward to improve the accuracy of the robot's angle position.

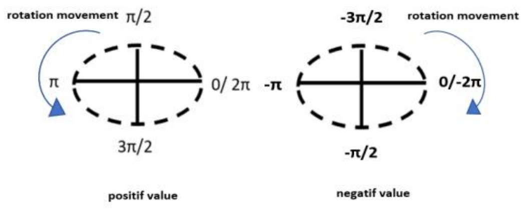

Figure 6 Angle of rotation for the mobile robot in this study

Essentially, a Lidar sensor detects obstacles and the surroundings to direct the robot's movement. In this work, 36 rays were applied for all training and simulations. The reason for adopting this small Lidar value is to assist the agent in evaluating and producing a minimum Lidar value for analysis. Furthermore, it can reduce the complexity of data processing.

D. Action Space

In this development, two types of action spaces are used for differential drive kinematics: 1. linear velocity and 2. Angular velocity. Since the action space(a_t) is a system output, so the neural network and system employed must also be selected based on the action that the robot will perform. In this research, a continuous action space is used to control the linear and angular velocity of the robot movement and ranges between -1 and 1 for the action, as mentioned in Equation (8).

$$a_t = \begin{cases} \textbf{linear velocity, with probability } -1 \leq a_t \leq 1 \\ \textbf{angular velocity, with probability } -1 \leq a_t \leq 1 \end{cases} \quad (8)$$

E. Reward Function

Our objective in this study is to investigate the reward function(r_t) and whether it would affect the result of the simulation. Previous articles [16, 24] have emphasized the importance of aligning the reward function with the mission requirements. Rewards are essential for motivating and assisting agents in efficiently completing tasks. In the realm of RL, successful outcomes are generously rewarded while poor performance is penalized. To explore different approaches, this research introduces four types of reward functions:

1. Reward function with angle correction (RFAC)
2. Reward function without angle correction (RFWAC)
3. Reward function without sparse reward (RFWSR)
4. Reward function without sparse reward and angle correction (RFWSRAC)

The development of the reward function is based on carefully analyzed criteria that are specifically tailored for the navigation task. Initially, the RFWSRAC incorporates the basic reward function for a mobile navigation robot, encompassing avoidance, distance, straight-line movement, and spinning movement. Subsequently, the RFWSR is enhanced by integrating angle correction into the RFWSRAC. Through observation, weaknesses related to exploitation learning were identified. To address these limitations, this research introduces the RFAC, which enhances the reward system by including an accomplishment reward based on sparse rewards. The RFAC incorporates avoidance, distance, straight-line movement, spinning movement, accomplishment, and angle correction as part of its reward function system. Finally, the RFWAC is designed without the angle correction reward to facilitate a comparison of its effectiveness with the comprehensive RFAC, which includes all the developed reward functions.

This section provides detailed explanations of each developed reward function and its purpose in providing effective rewards to guide the system towards achieving its objective in the navigation process, from the initial to the target position. The basic reward setup, utilizing the shaping reward function, involves incrementally rewarding the robot as it approaches the target position. On the other hand, the accomplishment reward function relies on sparse rewards, which are awarded when the robot successfully meets the criteria specified by the isDone condition (referring Table 4) for its position. The primary reward function is the distance reward, which calculates rewards based on the robot's proximity to the target. Additionally, five other fundamental criteria are used to assess the agent's performance. The avoidance reward assigns a positive value based on minimum distance by the Lidar robot to obstacle. The straight-line movement reward encourages the robot to move in a straight path based on linear velocity and penalizes spinning movements based on angular velocity. The accomplishment reward is the only sparse reward used in this research, and it is activated when the robot reaches the target pose as defined by the isDone condition. The agent receives an additional reward if it successfully reaches the target position. The angle correction reward is an additional function that provides rewards based on the robot's body positioning within a radius of less than 1 meter from the target position. It assigns higher rewards for movements that bring the robot closer to the target and penalizes movements away from the target. Table 2 depicts about the value for each reward function, and Table 3 describes the type of reward used in this development with its function.

Table 2 Value for each reward

Reward	**Value**			
	RFAC	**RFWAC**	**RFWSR**	**RFWSRAC**
Avoidance	0.01* R^2	0.01 * R^2	0.01 * R^2	0.01 * R^2
Straight line movement	1* v^2	1* v^2	1* v^2	1* v^2
Spinning movement	-1* ω^2	-1* ω2	-1* ω2	-1* ω2
Distance	1*D_T	1*D_T	1*D_T	1*D_T
Accomplish ment	P*2000	P*2000	not use	not use
Angle correction	1*E	not use	1*E	not use

where v is the linear velocity, ω is the angular velocity, R is the minimum range of Lidar, D_T is the total distance from target position and P is the accomplishment(termination) reward.

Table 3 Type of reward used with their function

Reward	Function	Type of Reward
Avoidance	Avoid nearest obstacle	Shaping
Straight line movement	Encourage straight line motion	Shaping
Spinning movement	Discourage going in circle	Shaping
Distance	Distance to target position	Shaping
Accomplishment	Accomplish to target position	Sparse
Angle correction	Encourage angle correction	Shaping

Based on Table 3, the rewards that are used, as well as the types of reward categories that represent different functions in guiding the agent to the target position are intended to demonstrate their effectiveness.

Equation (9) designates the reward function with angle correction (RFAC) with a combination of sparse (accomplishment reward) and shaping rewards. In contrast, Equation (10) designates the reward function without angle correction (RFWAC), Equation (11) designates the reward function without accomplishment reward (RFWSR), and Equation (12) designates the reward function without accomplishment reward and angle correction (RFWSRAC).

$$r(s_{t,}a_t) = \left(\frac{1}{100} \times R^2\right) + v^2 + (-\omega^2) + DT + P \times 2000 + E \quad (9)$$

$$r(s_{t,}a_t) = \left(\frac{1}{100} \times R^2\right) + v^2 + (-\omega^2) + DT + P \times 2000 \quad (10)$$

$$r(s_{t,}a_t) = \left(\frac{1}{100} \times R^2\right) + v^2 + (-\omega^2) + DT + E \quad (11)$$

$$r(s_{t,}a_t) = \left(\frac{1}{100} \times R^2\right) + v^2 + (-\omega^2) + DT \quad (12)$$

where v is the linear velocity, ω is the angular velocity, R is the minimum range of Lidar, D_T is the total distance from target position and P is the accomplishment reward.

Equation (13) is an extra condition to activate reward function with angle correction. The condition will be triggered if the agent arrives in a radius less than 1 meter before the target position, which encourages the agent to make a correction angle and to credit reward based on the positioning of the agent in the rotation movement of the robot's body.

$$AC = (\pi - (\sqrt{(E)^2})) * D_{ct} < 1\text{m} \quad (13)$$

where AC is an angle correction, E is an angle error condition, π is represented semi-circle of 360 degrees for the mobile robot movement and lastly D_{ct} is the distance between current to target position is defined as in Equation (5) and m is a meter.

In this experiment, the effectiveness of the reward system developed is crucial because each reward function has a significant impact on the robot's learning process when it interacts with the environment. For example, the accomplishment reward (sparse reward) is used to help the robot understand the importance of exploitation compared to exploration. It improves the quality of exploitation while reducing exploration, which is less important for learning. Exploitation is vital because the robot utilizes its knowledge to successfully complete tasks, whereas exploration encourages the robot to continuously learn and adapt to the environment based on the system's entropy value.

F. isDone

In navigation from the initial to the target position, the isDone function is used to accommodate the robot's boundary in determining the desired target. It allows the model to detect if the robot has successfully achieved the required task or failed to do so. There are four conditions to terminate the training process. The training episode ended when 1. The robot has arrived at to target location with tolerance, 2. The minimum range Lidar(robot) to wall or obstacle and any other object is crossed over, 3. Minimum and maximum theta (rotation position of the agent) and 4. The robot is reached maximum axis-x and axis-y. With this function, the system can stop the training if the robot has fulfilled the conditions. To determine several conditions, isDone uses a logic gate to execute containment action for the robot. Table 4 shows isDone condition used in this research.

Table 4 Type of isDone using in this development

Type of IsDone	Condition
Pose (target position with tolerance)	$c_x < (t_x \pm 0.5)$ AND $c_y < (t_y \pm 0.5)$ AND $\theta_c < (\theta_t \pm 0.25)$
Minimum Lidar to wall or obstacle	$c_L <$ min lidar distance (0.5m)
Minimum and maximum theta	$0 < \theta_c < 2\pi$ OR $0 > \theta_c > -2\pi$
Maximum x and y axis	only execute the running if the robot still exists on map

where c_x is the current axis-x, c_y is the current axis-y, t_x is the target axis-x, t_y is the target axis-y, θ_c is the current theta(θ), θ_t is the target theta(θ), c_L is the current Lidar range, AND is the AND logic gate and OR is the OR logic gate.

G. Environment Setup

The development of a two-dimensional (2D) map in MATLAB as per Figure 7 with the binary OccupancyMap method is used to create the robot's environment based on an occupancy grid with binary values. The dimensions used in this study for the training and simulation are 25 meters x 25 meters.

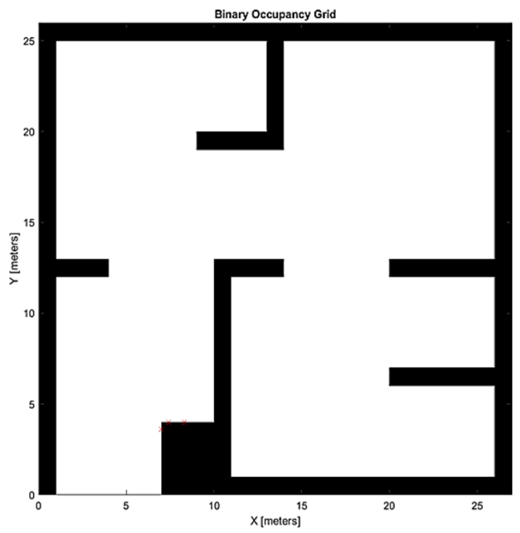

Figure 7 2D map using for the simulation

The parameters in the training process are shown in Table 5. Hyperparameter is a factor in determining the effectiveness of the system. The SAC algorithm allows the system maximum entropy for the discounted expected reward [14]. However, this framework is less sensitive to hyperparameter tuning consideration during the training process. The initial and target positions for the evaluation of the system are listed in Table 6.

Table 5 Hyperparameter in all simulations

Hyperparameter	Value
optimiser	Adam
learning rate	10^3
discount factor	0.99
replay buffer size	10^6
target smooth factor	10^3
mini batch size	256
sample time	0.1
episode maximum step	300

Table 6 Location of initial and target for the simulation

Axis	Position	
	Initial	Target
x	15	5 (±0.5)
y	15	5(±0.5)
θ	π /2	3π/2(±0.25)

H. Mobile Robot Model

One of the drives for a mobile robot is the use of differential drive kinematic to control movement. To determine the position of the mobile robot, three-element vectors (x, y, and theta(θ)) are used, with an x-y axis position in meters and a vehicle heading known as theta (θ) in radians. Time derivative states are determined by using the derivate function and the robot's current state. Furthermore, the derivative function is used to calculate the motion of a mobile robot with referring [37, 38]. The robot's movement will generate a pose, which will be used as the system's main observation. Figure 8 displays a mobile robot based on differential drive kinematics.

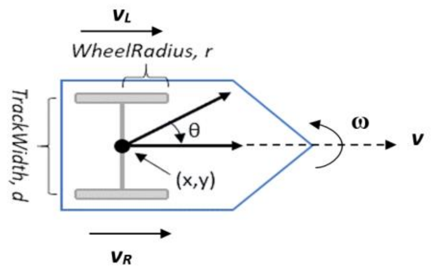

Figure 8 Mobile robot using differential drive kinematic

where x is the x-position for the robot, y is the y-position for the robot, *d* is the track width (in meters), θ(Theta) is the heading rotation(in radians), *r* is the wheel radius (in meters), *v* is the linear velocity (in meters/second), ω is the angular velocity (in meters/second), v_R is the velocity of the right wheel (in meters/second) and v_L is the velocity of the right wheel (in meters/second).

Meanwhile, a differential drive is determined by two factors are linear velocity (*v*) and angular velocity (ω). To define *v* and ω as described in Equation (14) – (15). The velocity of the robot is derived by taking the velocity of the right wheel (v_R) and the velocity of the left wheel (v_L).

$$\boldsymbol{v} = \frac{r}{2}(\boldsymbol{v_R}+\boldsymbol{v_L}) \tag{14}$$

then,

$$\boldsymbol{\omega} = \frac{r}{2d}(\boldsymbol{v_R}-\boldsymbol{v_L}) \tag{15}$$

M^T = [x, y, θ] ^{T} is a coordinate position based on a differential motion that can thus be defined following Equation (16) to determine the location and orientation of the robot movement. *T* denotes the current coordinates position.

$$M^T=\begin{bmatrix} x^T \\ y^T \\ \theta^T \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{16}$$

In the development of mobile robots, a differential-drive kinematic model based on a simple vehicle dynamic factor is used for simulation. Table 7 describes the properties used in this study.

Table 7 Properties used for Differential drive kinematic model

Parameters	Value
Input	Linear velocity(v) & Angular velocity(ω)
Wheel radius(m)	0.05
Wheel speed range(rad/s)	[-inf inf]
Track width (m)	0.2
Initial state	[initial X; initial Y; initial Theta]

3.2 Training and Simulation Setup

Firstly, we conduct four sets of reward function studies to investigate the reliability of the performance based on learning accuracy, as shown in Table 2. This experiment has four reward functions, RFAC, RFWAC, RFWSR, and RFWSRAC. The training process will record in 1000 episodes. The training data will be collected and compared for each reward function. During training, the agent will learn and explore an environment where the system will record all reward-related data and the number of steps for analysis. After the training, the agent will be saved and used to evaluate their performance with the same initial and target position. For the simulation, each movement of the agent will be recorded to validate the reward function's efficiency and accuracy. To perform experiments with four different reward functions, with all parameters remaining constant for both reward function systems, including the system's input and output.

Furthermore, we analysed the collected data and evaluated the result based on the agent's ability to navigate from the initial to the target position. To ensure that our study's objectives are achieved, we propose two approaches: 1. complete the training process, and 2. evaluate the agent trained with the simulation.

4.0 RESULTS AND DISCUSSION

To achieve our objective of the study, the result of the simulation can determine whether the system developed to deliver our objective or target.

4.1 Training Result

In this study, the training of the agent in interaction with the environment is determined by two aspects: average reward, and step of movements.

The average reward is displayed in Figure 9. Overall, these results indicate that there is a clear tendency for a similar pattern. Early in the learning process, the agent will have a high risk of exploration due to fluctuating rewards and getting bad result. After the 700th episode, it can see that the agent started to adapt and mature in an environment to get higher rewards. Based on the average result reward for RFWSR takes longer to stable compared with RFAC and RFWAC. This result is most likely related to the reward function, where RFWSR stands for reward function without sparse reward (accomplishment reward). Basically, the sparse reward is used as a containment action to reduce unnecessary exploration and provide direct results if their goal is achieved. Meanwhile, RFWSRAC probably suffers in learning because it lacks the accomplishment and angle correction reward functions. All reward functions are fast matured after the 700th episode refer to Figure 9 where the reward is more stable and consistence. However, it must measure the system's maturity based on learning accuracy.

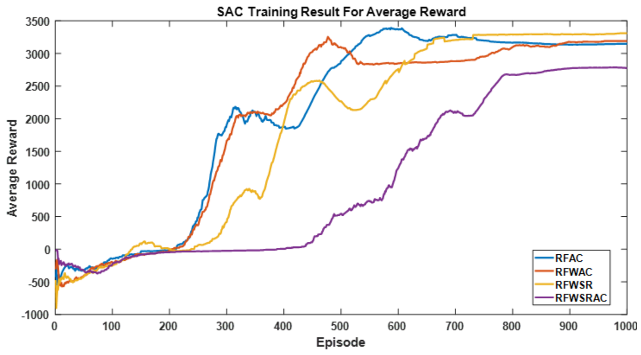

Figure 9 The result of training for average reward

Figure 10 present the average steps for 1000 episodes of the training. Another possible explanation for this is that exploration demonstrates the instability of the use of maximum steps induced by the obstacle wall, making it more difficult for the agent to explore the environment during an early stage due to the narrower space. Then, the agent starts understanding and adapting to the environment based on the increment of the steps. In comparison, RFAC and RFWAC use steps less after the 700th episode, whereas RFWSR and RFWSRAC use maximum steps at the end of the training after the 700th episode. A possible explanation for this might be that RFAC and RFWAC are integrated with sparse reward into the reward function system, whereas RFWSR and RFWSRAC are not.

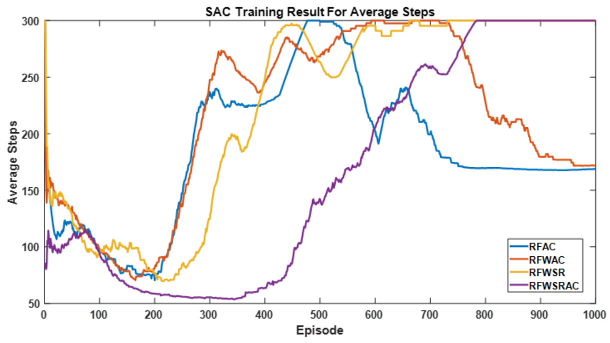

Figure 10 The result of training for average steps

4.2 Simulation Result

The simulation result is produced by a validated system after the training session, where the agent is used to display behaviours of the agent. A comparison of distance error and error percentage can indicate an agent's accuracy in navigating to the desired location. To determine distance error percentage as following Equation (17).

$$Error\,\% = \frac{|tv - av|}{tv} \times 100 \tag{17}$$

where tv is the target value and av is the arrived value.

To make it more straightforward to evaluate system performance, an average percentage to determine the best overall system is used. That is the average error percentage in the Equation (18).

$$AE\,\% = \frac{(s1\% + s2\% + s3\%)}{n} \tag{18}$$

where AE% is average error percentage, s1% is the error percentage vector-x, s2% is the error percentage vector-y, s3% is the error percentage vector-theta and n is number of samples.

According to Table 8, the smaller the distance error or error %, the more accurate the agent's position. The simulation process shows that RFAC performs significantly better than RFWAC, RFWSR, and RFWSRAC based on average error percentage. RFAC obtained a value of 3.9% for the x-axis, 9.54% for the y-axis, 1.53% for theta, and an average error of 4.99%. Although RFWAC recorded 5.77% is the second best with percentage positions of 4.42%, 9.8%, and 3.08%. Moreover, RFWSRAC had a high average error percentage of 99.25%, followed by RFWSR, which had a percentage of 63.26%. In contrast, the worst results are divided into the x, y-axis, and theta position. Regarding the Theta position, RFWSRAC is the lowest with a value of 234.68%, while RFWSR is 21.75%. Then, on the x, y-axis, and RFWSR has the highest error % of 70%, 98.04%, while RFWSRAC has a slightly lower error % of 17.74%, and 45.34%.

Table 8 The result of simulation

Simulation	Axis	Position			
		Final position	Distance error (m/ radians)	Error %	Average error %
RFAC	x	5.2	0.2	3.9	4.99
	y	5.48	0.48	9.54	
	θ	4.64	0.07	1.53	
RFWAC	x	5.22	0.22	4.42	5.77
	y	5.49	0.49	9.8	
	θ	4.57	0.15	3.08	
RFWSR	x	1.5	3.5	70.00	63.26
	y	0.1	4.9	98.04	
	θ	3.69	1.03	21.75	
RFWSRAC	x	4.11	0.89	17.74	99.25
	y	2.73	2.27	45.34	
	θ	-6.35	11.06	234.68	

Through movement trajectories categorized as x and y-axis in Figure 11 and theta rotation position in Figure 12. In terms of accuracy, RFAC and RFWAC outperform RFWSR and RFWSRAC (see Figure 11). Furthermore, RFAC and RFWAC movements are highly similar. RFWSR and RFWSRAC have lower accuracy, implying the importance of accomplishment reward (sparse reward) in helping the system learn more efficiently and quality.

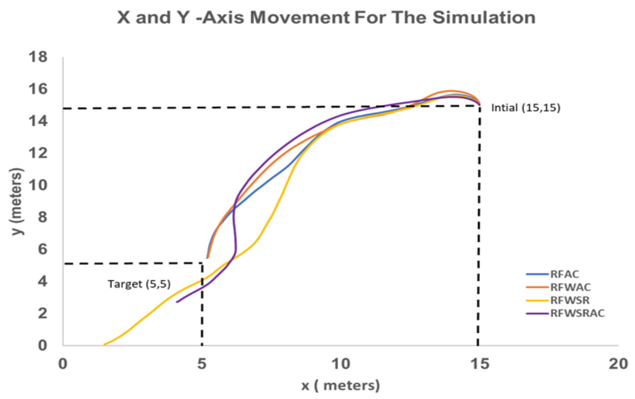

Figure 11 Movement of the agent based on the x and y-axis for the simulation

The difference in the agent movement pattern for the rotation angle is shown in Figure 12. RFAC and RFWAC are almost marginal, but the difference between RFWSR and RFWSRAC is quite different. RFAC is almost accurate in rotation position, whereas RFWSRAC is far from the target rotation angle due to the negative value. The factor affecting the movement rotation in RFWSRAC is the non - availability of an angle correction reward to motivate in acquiring rotation accuracy and no accomplishment reward, which makes the learning process more difficult.

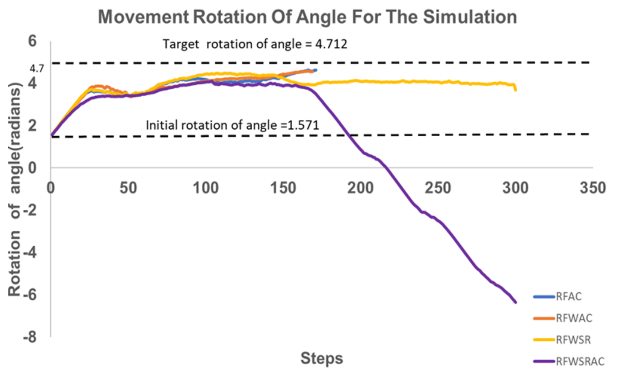

Figure 12 Movement rotation of angle for the simulation

Figure 13 indicates the simulation result for all reward functions. RFAC and RFWAC shows that the agent was successful in navigating from an initial to a target location with tolerance, which is also can refer to in Table 6 & 8. However, RFWSR shows the worst movement due to a far from the target position until it hits the limitation of the map, as shown in Figure 13. The results of the RFWSRAC simulation, where the result is not accurate with more marginal values on the x- and y-axis. Still, the error percentage is high for rotation based on body rotation is not an accurate position due to incorrect turning rotation, and it produces a negative value.

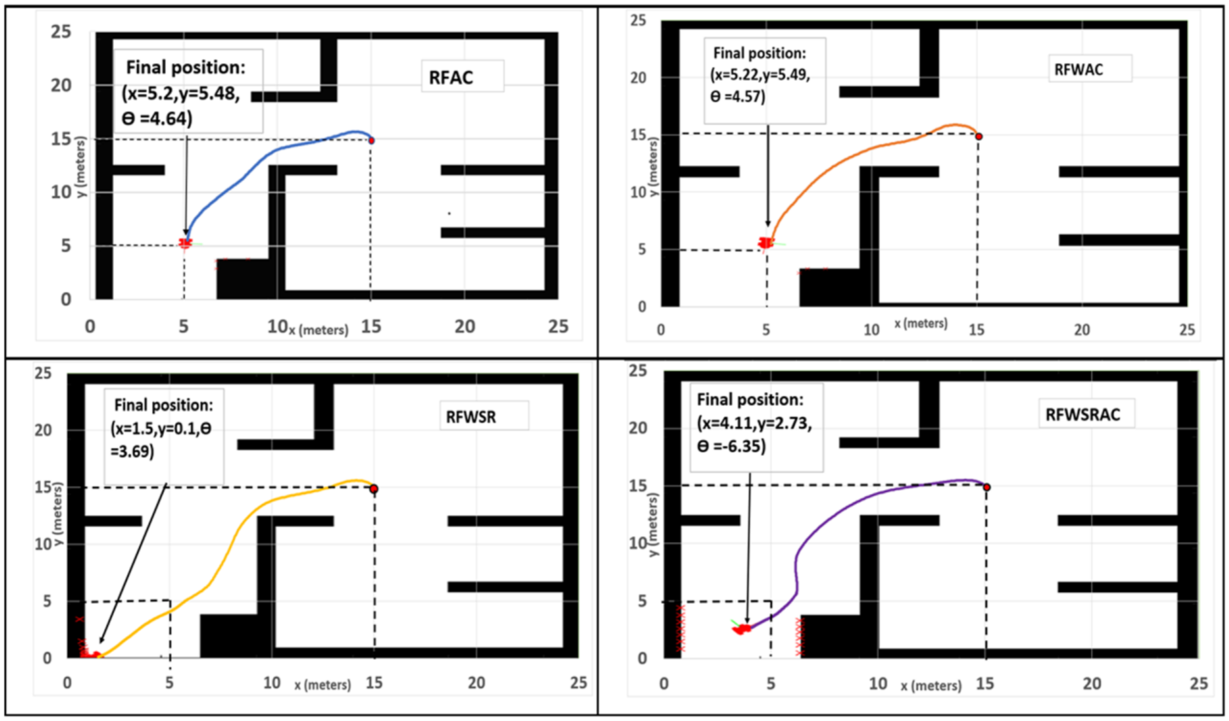

Figure 13 The result of the simulation for all reward functions

These experiments demonstrated agents successfully navigating from the start to the target location using two reward functions, which are RFAC and RFWAC with tolerance (as referred to in Table 6) but not RFWSR and RFWSRAC. However, as shown in the robot's accuracy in reaching the target position is vital in determining the effectiveness of the developed reward system. Through this experiment, the result of the training and simulation systems that employ appropriate reward strategies especially, RFAC can assist robots in learning and adapting to their environment more efficiently and quickly mature. The fusion of the sparse and shaping reward has been demonstrated through this study and successfully improved the agent's accuracy and maturity.

This study shows that using a good reward function can help stimulate the robot to interact more effectively, where the factors of exploration and exploitation needs to be considered. The shaping reward will help improve the exploration process by guiding action to the objective. The sparse reward can reward the process if it is successful based on binary system reward {true-1, false -0}. This study demonstrated that shaping reward is beneficial in the exploration process through this experiment, especially distance reward influences the robot to the target position. For sparse rewards, which will reward if the system succeeds in achieving the goal and help the system learn more easily and it is very effective in exploitation learning.

Commonly, the SAC algorithm is applied to solve problems involving high-dimensional state action for continuous action[14], as demonstrated in the results of our experiments in Figure 13, which successfully navigate the robot to the initial to target position, but the results are different due to the use of different reward function systems. Based on the previous article emphasising the use of the reward function in the Reinforcement learning system, [24] conducted an experiment based on the researcher's knowledge domain that uses the shaping reward in assisting the learning system. In contrast, this study introduces a fusion of shaping and sparse reward in helping to accelerate the maturation of robot learning based on a stable learning process by referring to Figure 9, where it matured after the 700th episode. Furthermore, the development of this fusion can improve learning performance based on target position accuracy (refer to Table 8). This article [6] uses only shaping rewards for reward function systems in SAC and DDPG methods but requires the 7000th episode to train the learning robot. Compared to this study, the robot successfully learns with high accuracy and matures quickly to navigate to the target location with only one thousand training episodes based on the reward function that uses fusion sparse and shaping reward with refer to RFAC.

Finally, the conclusion is that the reward system must meet the requirements and that the agent must be able to self-motivate to achieve the objective system. For example, a person that receives appropriate encouragement or incentive to motivate himself will learn more effectively.

4.0 CONCLUSION

This research is motivated by implementing fusion sparse and shaping rewards in SAC DRL for mobile robot navigation. To enable the system to learn effectively, using the reward function appropriately and precisely assists the system in learning accuracy. Based on the results of the experiment, RFAC produces a low average error value of 4.99%, proving that the employment of fusion sparse and shaping reward in SAC increases learning accuracy and maturity. However, reward function without sparse reward, such as RFWSR and RFWSRAC, produce high average error values of 63.26% and 99.25%, respectively.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

Acknowledgement

This research was supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2022/ICT02/UTHM/03/1), Robocup Federation - MathWorks Support for Research Projects 2021, Research Management Centre (RMC) Universiti Tun Hussein Onn Malaysia (UTHM), Faculty of Electrical and Electronic Engineering (FKEE) of UTHM and Universiti Tun Hussein Onn Malaysia (UTHM). Finally, the author would like to acknowledge MathWorks as a collaboration partner, especially in technical support for this research.

References

[1] X. Yang, M. Moallem, and R. V. Patel. 2005. A Layered Goal-oriented Fuzzy Motion Planning Strategy for Mobile Robot Navigation. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* 35(6): 1214-1224. Doi: 10.1109/TSMCB.2005.850177.

[2] M. Faisal, M. Algabri, B. M. Abdelkader, H. Dhahri, and M. M. Al Rahhal. 2017. Human Expertise in Mobile Robot Navigation. *IEEE Access*. 6: 1694-1705. Doi: 10.1109/ACCESS.2017.2780082.

[3] M. P. Deisenroth. 2013. A Survey on Policy Search for Robotics. *Found. Trends® Robot.* 2(1-2): 1-142. Doi: 10.1561/2300000021.

[4] A. S. Polydoros and L. Nalpantidis. 2017. Survey of Model-Based Reinforcement Learning: Applications on Robotics. *J. Intell. Robot. Syst. Theory Appl.* 86(2): 53-173. Doi: 10.1007/s10846-017-0468-y.

[5] J. Xiang, Q. Li, X. Dong, and Z. Ren. 2019. Continuous Control with Deep Reinforcement Learning for Mobile Robot Navigation. *Proc. - 2019 Chinese Autom. Congr. CAC 2019*. 1501-1506. Doi: 10.1109/CAC48633.2019.8996652.

[6] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. de S. L. Cuadros, and D. F. T. Gamarra. 2021. Soft Actor-Critic for Navigation of Mobile Robots. *J. Intell. Robot. Syst. Theory Appl.* 102(2): 1-11. Doi: 10.1007/S10846-021-01367-5/METRICS.

[7] G. Chen *et al.* 2020. Robot Navigation with Map-Based Deep Reinforcement Learning. *2020 IEEE Int. Conf. Networking, Sens. Control. ICNSC 2020*. Doi: 10.1109/ICNSC48988.2020.9238090.

[8] R. N. Das, K., & Behera. 2017. A Survey On Machine Learning: Concept, Algorithms and Applications. *Int. J. Innov. Res. Comput. Commun. Eng.* 5(2): 1301-1309. Doi: 10.15680/IJIRCCE.2017.

[9] S. Irfan, A. Meerza, M. Islam, and M. M. Uzzal. 2019. Q-Learning Based Particle Swarm Optimization Algorithm for Optimal Path Planning of Swarm of Mobile Robots. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. Doi: 10.1109/ICASERT.2019.8934450.

[10] X. Luo, Y. Gao, S. Huang, Y. Zhao, and S. Zhang. 2019. Modification of Q-learning to Adapt to the Randomness of Environment. *2019 Int. Conf. Control. Autom. Inf. Sci.* 1-4. Doi: 10.1109/ICCAIS46528.2019.9074718.

[11] C. S. Arvind and J. Senthilnath. 2019. Autonomous RL: Autonomous Vehicle Obstacle Avoidance in a Dynamic Environment using MLP-SARSA Reinforcement Learning. *2019 IEEE 5th Int. Conf. Mechatronics Syst. Robot. ICMSR 2019*. 120-124. Doi: 10.1109/ICMSR.2019.8835462.

[12] V. Mnih *et al.* 2015. Human-level Control through Deep Reinforcement Learning. *Nature*. 518(7540): 529-533. Doi: 10.1038/nature14236.

[13] Y. Wang, J. Tong, T. Y. Song, and Z. H. Wan. 2018. Unmanned Surface Vehicle Course Tracking Control based on Neural Network and Deep Deterministic Policy Gradient Algorithm. *2018 Ocean - MTS/IEEE Kobe Techno-Oceans, Ocean - Kobe 2018*. Doi: 10.1109/OCEANSKOBE.2018.8559329.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *PMLR*. 1861-1870. Accessed: Feb. 14, 2023. [Online]. Available: https://proceedings.mlr.press/v80/haarnoja18b.html.

[15] R. S. Sutton and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

[16] X. Yu, Y. Sun, X. Wang, and G. Zhang. 2021. End-to-End AUV Motion Planning Method Based on Soft Actor-Critic. *Sensors.* 21(17): 5893. Doi: 10.3390/S21175893.

[17] R. Takehara and T. Gonsalves. 2021. Autonomous Car Parking System using Deep Reinforcement Learning. *2nd Int. Conf. Innov. Creat. Inf. Technol. ICITech 2021*. 85-89. Doi: 10.1109/ICITECH50181.2021.9590169.

[18] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li. 2020. Deep Interactive Reinforcement Learning for Path Following of Autonomous Underwater Vehicle. *IEEE Access*. 8: 24258-24268. Doi: 10.1109/ACCESS.2020.2970433.

[19] C. Wu *et al.* 2019. UAV Autonomous Target Search based on Deep Reinforcement Learning in Complex Disaster Scene. *IEEE Access*. 7: 117227-117245. Doi: 10.1109/ACCESS.2019.2933002.

[20] K. Zhu and T. Zhang. 2021. Deep Reinforcement Learning based Mobile Robot Navigation: A Review. *Tsinghua Sci. Technol.* 26(5): 674-691. Doi: 10.26599/TST.2021.9010012.

[21] Mohamad Hafiz Abu Bakar, Abu Ubaidah bin Shamsudin, Ruzairi Abdul Rahim, Zubair Adil Soomro, and Andi Adrianshah. 2023. Comparison Method Q-Learning and SARSA for Simulation of Drone Controller using Reinforcement Learning. *J. Adv. Res. Appl. Sci. Eng. Technol.* 30(3 SE-Articles): 69-78. Doi: 10.37934/araset.30.3.6978.

[22] L. Meng, R. Gorbet, and D. Kulić. 2020. The Effect of Multi-step Methods on Overestimation in Deep Reinforcement Learning. *Proc. - Int. Conf. Pattern Recognit.* 9740-9747. Doi: 10.1109/ICPR48806.2021.9413027.

[23] Z. Yang, K. Merrick, S. Member, L. Jin, H. A. Abbass, and S. Member. 2018. Hierarchical Deep Reinforcement Learning for Continuous Action Control. *IEEE Trans. neural networks Learn. Syst.* 29(11): 5174-5184.

[24] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan. 2019. Deep Reinforcement Learning with Optimized Reward Functions for Robotic Trajectory Planning. *IEEE Access*. 7: 105669-105679. Doi: 10.1109/ACCESS.2019.2932257.

[25] J. Hare. 2019. *Dealing with Sparse Rewards in Reinforcement Learning*. arXiv:1910.09281.

[26] C. Wang, J. Wang, J. Wang, and X. Zhang. 2020. Deep-Reinforcement-Learning-based Autonomous UAV Navigation with Sparse Rewards. *IEEE Internet Things J.* 7(7): 6180-6190. Doi: 10.1109/JIOT.2020.2973193.

[27] M. Riedmiller *et al.* 2018. Learning by Playing - Solving Sparse reward Tasks from Scratch. *35th Int. Conf. Mach. Learn. ICML 2018*. 10: 6910-6919.

[28] C. Wang, J. Wang, Y. Shen, and X. Zhang. 2019. Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach. *IEEE Trans. Veh. Technol.* 68(3): 2124-2136. Doi: 10.1109/TVT.2018.2890773.

[29] Y. Hu, Y. Hua, W. Liu, and J. Zhu. 2021. Reward Shaping based Federated Reinforcement Learning. *IEEE Access*. 9: 67259-67267. Doi: 10.1109/ACCESS.2021.3074221.

[30] A. Laud and G. DeJong. 2003. The Influence of Reward on the Speed of Reinforcement Learning: An Analysis of Shaping. *Proceedings, Twent. Int. Conf. Mach. Learn.* 1: 440-447.

[31] W. Wang, Z. Wu, H. Luo, and B. Zhang. 2022. Path Planning Method of Mobile Robot Using Improved Deep Reinforcement Learning. *J. Electr. Comput. Eng.* Doi: 10.1155/2022/5433988.

[32] A. Trott, S. Research, S. Zheng, C. Xiong, and R. Socher. 2019. Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards. *Adv. Neural Inf. Process. Syst.* 32.

[33] A. Hussein, E. Elyan, M. M. Gaber, and C. Jayne. 2017. Deep Reward Shaping from Demonstrations. *Proc. Int. Jt. Conf. Neural Networks*. 510-517. Doi: 10.1109/IJCNN.2017.7965896.

[34] Soft Actor-Critic Agents - MATLAB & Simulink. https://www.mathworks.com/help/reinforcement-learning/ug/sac-agents.html (accessed Feb. 15, 2023).

[35] TensorFlow for Deep Learning [Book]. https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ (accessed Feb. 15, 2023).

[36] A. D. Rasamoelina, F. Adjailia, and P. Sincak. 2020. A Review of Activation Function for Artificial Neural Network. *SAMI 2020 - IEEE 18th World Symp. Appl. Mach. Intell. Informatics, Proc.* 281-286. Doi: 10.1109/SAMI48414.2020.9108717.

[37] K. M. Lynch and F. C. Park. 2017. *Modern Robotics*. Cambridge University Press.

[38] Mobile Robot Kinematics Equations - MATLAB & Simulink. https://www.mathworks.com/help/robotics/ug/mobile-robot-kinematics-equations.html (accessed Aug. 23, 2023).